- Master theorem
- Quick Sort

$T(n) = 2T(\frac{n}{2}) + O(n)$

Master theorem

Recurrence: $T(n) = aT(\frac{n}{b}) + f(n)$

$a \geq 1$, $b > 1$, $f(n)$: function of n

3 cases: Compare $f(n)$ vs $n^{\log_b a}$

Case 1: if $f(n) = O(n^{\log_b a - \varepsilon})$

$T(n) = \Theta(n^{\log_b a})$

Case 2: if $f(n) = \Theta(n^{\log_b a})$

$T(n) = \Theta(n^{\log_b a} \cdot \log n)$

ε: some positive constant

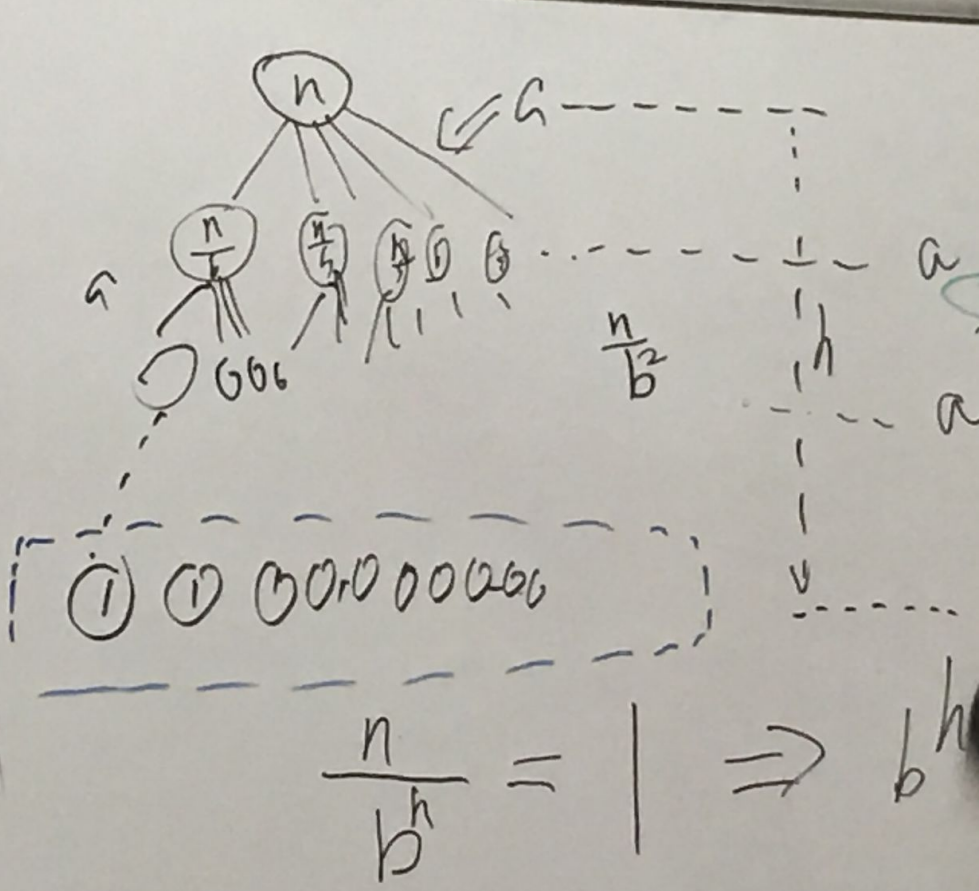
Case 3: if $f(n) = \Omega(n^{\log_b a + \varepsilon})$

also $f(n)$ satisfy regularity condition:

$$a f\left(\frac{n}{b}\right) \leq c \cdot f(n)$$ for large n

(a, b: constant; c: constant)

$$T(n) = \Theta(f(n))$$



Ex:

① $T(n) = 2T(\frac{n}{2}) + n$

$a = 2, \; b = 2, \; n^{\log_b a} = n$

$f(n) = n$

$f(n) = \Theta(n^{\log_b a})$ case 2

$n = \Theta(n)$

$n = \Theta(n^{1-\varepsilon})$

$n = \Theta(n^{\log_b a - \varepsilon})$

$T(n) = \Theta(n \cdot \log n)$

② $T(n) = 9T(\frac{n}{3}) + n$

$a = 9, \; b = 3$

$n^{\log_b a} = n^2$

$f(n) = n$

$= O(n^{2-\varepsilon})$ for $\varepsilon = 1$

$T(n) = \Theta(n^2)$

(3) $T(n) = T\left(\frac{2}{3}n\right) + 1$

$a = 1, \quad b = \frac{3}{2}, \quad \log_b a = 0$

$n^{\log_b a} = n^0 = 1$

$f(n) = 1 = \Theta(n^{\log_b a} \cdot \log^2 n)$

$T(n) = \Theta(1 \cdot \log n) = \Theta(\log n)$

(4) $T(n) = 3T(\frac{n}{4}) + n\log n$

$a = 3, \quad b = 4, \quad \log_b a = 0.793$

$f(n) = n\log n = \Omega(n^{0.793+\varepsilon}) \Rightarrow 1$

$\varepsilon = 0.207$

$n^{\log_b a}$

n

$f(n) = n \log n$

④ $T(n) =$

$af\left(\frac{n}{b}\right) = 3 \cdot \left[\frac{n}{4} \log \frac{n}{4}\right) \leq C \cdot n \log n$

$a =$

$f(n) =$

$= 3 \cdot f\left(\frac{n}{4}\right) = \frac{3}{4} n [\log n - \log 4]$

Let $C =$

$= \frac{3}{4} n \log n - \frac{3}{4} n \cdot \log 4 \leq C \cdot n \log n$

(5) $T(n) = 2T(\frac{n}{2}) + n\log n$

$a = 2, \quad b = 2$

$n^{\log_b a} = n^{\log_2 2} = n^1 = n$

$f(n) = n\log n = \Omega(n^{1+\epsilon})$

$\log n = \Omega(n^{1.01})$

$\log n = \Omega(n^{0.01})$

$f(n) = n\log n$

$af(\frac{n}{b}) = 3 \cdot [\frac{n}{4} \cdot \log\frac{n}{4}]$

$= 3 \cdot f(\frac{n}{4}) = \frac{3}{4} n(\log n - \log 4)$

$= \frac{3}{4} n\log n - \frac{3}{4} n \cdot$

(6) $T(n) = T\left(\frac{n}{2}\right) + 1$

$a = 1, \quad b = 2 \quad n^{\log_b a} = n^0 = 1$

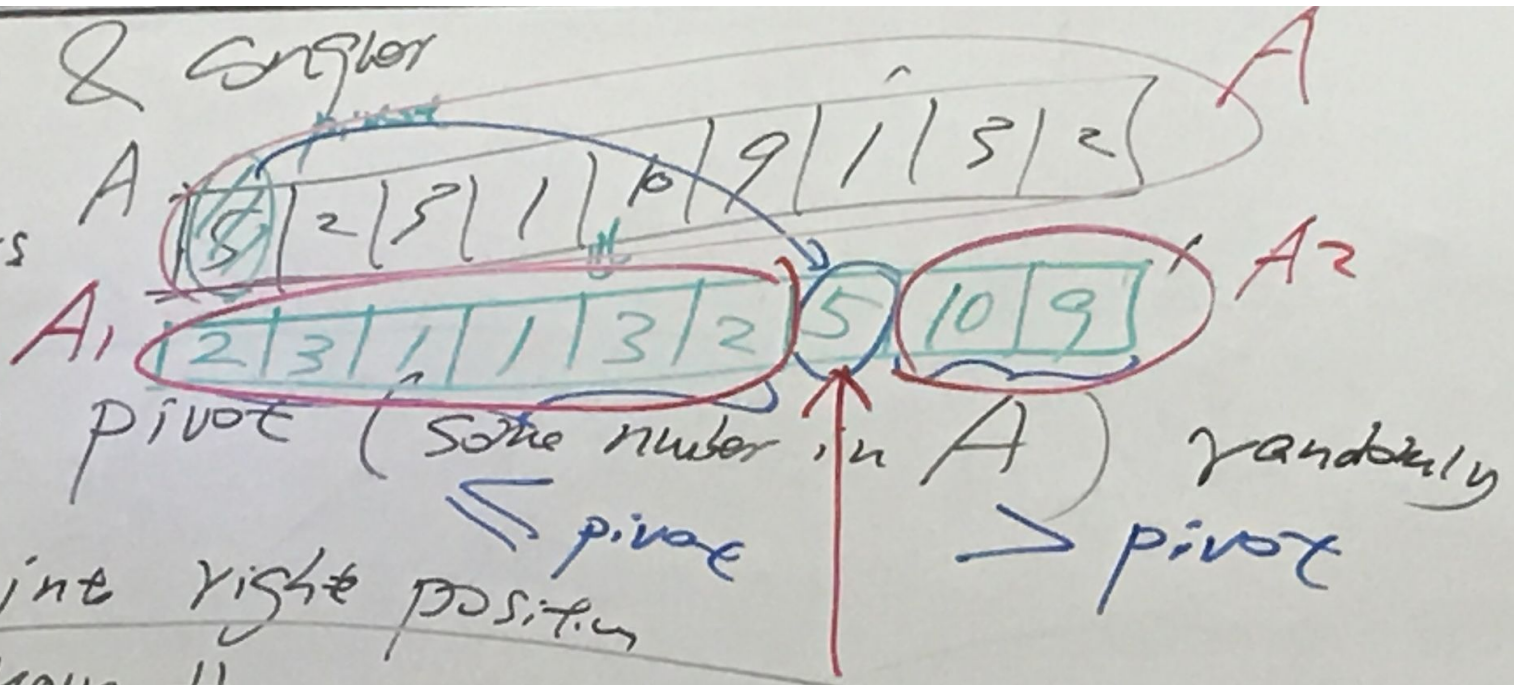
$f(n) = 1 = \Theta(n^{\log_b a})$

Case 2:

$T(n) = \Theta(1 \cdot \log n) = \Theta(\log n$

Quicksort (A) divide & conquer

List: A of numbers



① Divide: pick a pivot (some number in A) randomly

place pivot into right position

and move # ≤ pivot before pivot

..... > .... after

A1

A2

② Conquer: Quicksort (A1)

Quicksort (A2)

Time:

① Worse case.

$T(n) = T(n-1) + O(n)$ ($O(n) = c \cdot n$)

$T(n) = T(n-1) + cn$

$= [T(n-2) + c(n-1)] + cn$

$= [T(n-3) + c(n-2) + c(n-1)] + cn$

$\vdots$

$= T(1) + c \cdot (2) + c \cdot 3 + \cdots + c \cdot (n-1) + cn$

$= T(1) + c \cdot [2 + 3 + 4 \cdots + n]$

$= T(1) + c \cdot \left[\frac{n(n+1)}{2} - 1\right) = \Theta(n^2)$

② $T(n) = 2T\left(\frac{n}{2}\right) + O(n)$

$T(n) = O(n \log n)$

① ② $T(n) = 9T\left(\frac{n}{3}\right) + n$

$a = 9,\ b = 3$ $n^{\log_b a} = n^2$

$f(n) = n$

$= O(n^{2-\varepsilon})$ for $\varepsilon = 1$

$T(n) = \Theta(n^2)$

CSE 3500 Algorithms and Complexity – Fall 2016
Lecture 8: September 22, 2016

Master Theorem

- In the last lecture we showed how to use the repeated substitutions technique to solve a recurrence relation. In this lecture we study the Master theorem.
- The Master theorem considers a recurrence relation of the kind:

$$T(n) = \begin{cases} c & \text{if } n \leq d \\ aT(n/b) + f(n) & \text{if } n > d \end{cases}$$

  where $a > 0, b > 1, c$, and d are integer constants, and $f(n)$ is a non-negative integer function of n.
- There are three cases to consider and in each case we compare $n^{\log_b a}$ with $f(n)$.
- **Case 1:** $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$. In this case the solution is: $T(n) = \Theta(n^{\log_b a})$.
- **Case 2:** $n^{\log_b a} = \Theta(f(n))$. In this case the solution is: $T(n) = \Theta(f(n) \log n)$.
- **Case 3:** $n^{\log_b a} = O(f(n)/n^{\epsilon})$ for some constant $\epsilon > 0$ and there exists a constant $q < 1$ such that $af(n/b) \leq qf(n)$. In this case, $T(n) = \Theta(f(n))$.

Example 1

- The recurrence relation for binary search we obtained was:

$$T(n) = \begin{cases} 1 & \text{if } n \leq 1 \\ T(n/2) + 1 & \text{if } n > 1 \end{cases}$$

- For this recurrence relation $a = 1, b = 2$ and $f(n) = 1$. $n^{\log_b a} = n^{\log_2 1} = 1$. Thus, $f(n) = \Theta(n^{\log_b a}$. As a result, case 2 holds and we infer that $T(n) = \Theta(f(n) \log n) = \Theta(\log n)$.

Example 2

- The recurrence relation for merge sort is:

$$T(n) = \begin{cases} 1 & \text{if } n \leq 2 \\ 2T(n/2) + n & \text{if } n > 2 \end{cases}$$

- For this recurrence relation $a = 2, b = 2$ and $f(n) = n$. $n^{\log_b a} = n^{\log_2 2} = n$. Thus, $f(n) = \Theta(n^{\log_b a}$. As a result, case 2 holds and we infer that $T(n) = \Theta(f(n) \log n) = \Theta(n \log n)$.

Example 3

- Consider the following recurrence relation:

$$T(n) = \begin{cases} 1 & \text{if } n \leq 2 \\ 5T(n/2) + n^2 & \text{if } n > 2 \end{cases}$$

- For this recurrence relation $a = 5, b = 2$ and $f(n) = n^2$. $n^{\log_b a} = n^{\log_2 5}$. Note that $\log_2 5 > 2.32$. Thus, for a value of $\epsilon = 0.3$, $f(n) = O(n^{\log_b a - \epsilon})$. This means that case 1 holds and hence $T(n) = \Theta(n^{\log_2 5})$. This is what we got using the repeated substitutions technique as well. Note that the choice of ϵ is not unique. For example, we could have chosen ϵ to be 0.2.

Example 4

- Now consider the following recurrence relation:

$$T(n) = \begin{cases} 1 & \text{if } n \leq 2 \\ 12T(n/3) + n^3 & \text{if } n > 2 \end{cases}$$

- For this recurrence relation $a = 12, b = 3$ and $f(n) = n^3$. $n^{\log_b a} = n^{\log_3 12}$. Note that $\log_3 12 < 2.3$. Thus, for a choice of $\epsilon = 0.5$, $n^{\log_b a} = O\left(\frac{f(n)}{n^{0.5}}\right)$. Also, $a\ f(n/b) = 12(n/3)^3 = (12/27)n^3$. In other words, $a\ f(n/b) \leq qf(n)$ for $q = (12/27)$ which is less than one. As a result, case 3 holds. Therefore, we conclude that $T(n) = \Theta(n^3)$.

Quick sort

- Input: $X = k_1, k_2, \ldots, k_n$; Output: Sorted X.

- The quick sort algorithm employs divide-and-conquer and works as follows:

QuickSort(X)
  if $n = 1$ **then** quit;
  if $n = 2$ **then**
    {**if** $k_1 > k_2$ **then** swap k_1 and k_2; quit;}
  Pick a pivot element k from X;
  Partition X into X_1 and X_2 using k as follows:
    $X_1 = \{q \in X : q < k\}$ and $X_2 = \{q \in X : q > k\}$;
  Recursively sort X_1 to get Y_1;
  Recursively sort X_2 to get Y_2;
  Output Y_1, k, Y_2;

Analysis of quick sort

- Let $T(n)$ be the run time of quick sort on any input of size n.
- Given that partitioning takes n comparisons, we see that: $T(n) = T(|X_1|)+T(|X_2|)+n$.
- One of the worst cases happens when the input is already in sorted order. In this case, whenever a recursive call is made, one of the two parts is empty.
- The recurrence relation for $T(n)$ corresponding to the above worst case is: $T(n) = T(n-1) + n = T(n-2) + (n-1) + n = T(n-3) + (n-2) + (n-1) + n = \cdots = \sum_{i=1}^{n} i = \frac{n(n+1)}{2} = \Theta(n^2)$.
- Consider the possibility that whenever a recursive call is made, both X_1 and X_2 are of the same size. In this case the recurrence relation for $T(n)$ will become: $T(n) = 2T\left(\frac{n}{2}\right)+n$. This solves to: $T(n) = \Theta(n \log n)$ (using the Master theorem, for example). This is one of the best cases.
- In the next lecture we will show that the expected run time of quick sort is $O(n \log n)$.